

(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局(43) 国際公開日
2004 年 3 月 25 日 (25.03.2004)

PCT

(10) 国際公開番号
WO 2004/025468 A1(51) 国際特許分類⁷: G06F 11/00, 9/32, 9/34, H03K 19/177

(21) 国際出願番号: PCT/JP2002/009437

(22) 国際出願日: 2002 年 9 月 13 日 (13.09.2002)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人 (米国を除く全ての指定国について): 株式会社日立製作所 (HITACHI, LTD.) [JP/JP]; 〒101-8010 東京都千代田区神田駿河台四丁目6番地 Tokyo (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 田中 博志 (TANAKA, Hiroshi) [JP/JP]; 〒185-8601 東京都国分寺

市 東恋ヶ窪一丁目280番地 株式会社日立製作所中央研究所内 Tokyo (JP).

(74) 代理人: 作田 康夫 (SAKUTA, Yasuo); 〒100-8220 東京都千代田区丸の内一丁目5番1号 株式会社日立製作所内 Tokyo (JP).

(81) 指定国 (国内): CN, JP, KR, US.

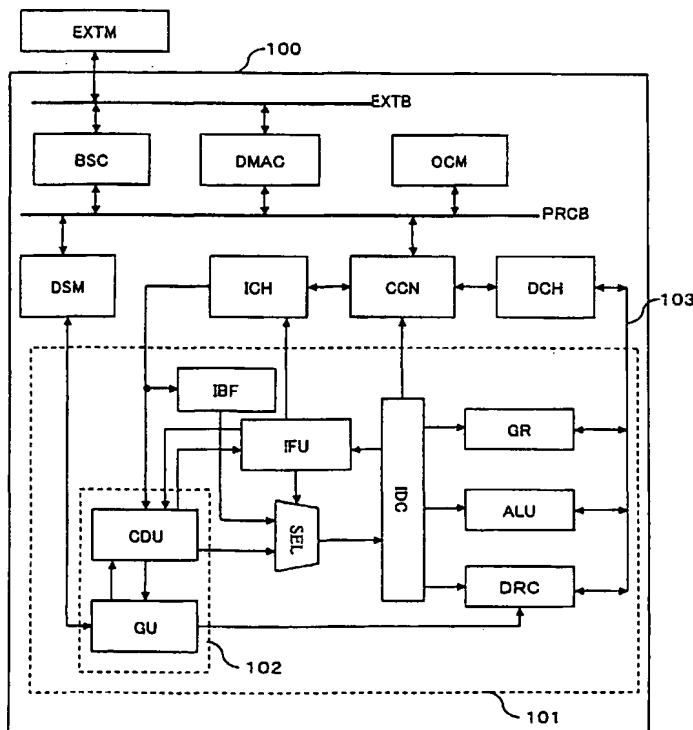
(84) 指定国 (広域): ヨーロッパ特許 (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR).

添付公開書類:
— 国際調査報告書

2文字コード及び他の略語については、定期発行される各PCTガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

(54) Title: SEMICONDUCTOR DEVICE

(54) 発明の名称: 半導体装置



(57) Abstract: A semiconductor device having a circuit which can be rewritten dynamically and a function for maintaining software compatibility in spite of the dynamically rewritable configuration. Simultaneously with software execution, data for rewriting a dynamically rewritable circuit and driver software for operating this are automatically generated and the latter part of an original program is replaced by the generated data. Thus, by maintaining software compatibility and diverting an existing software resource, it is possible to use the same software in various devices.

(続葉有)



(57) 要約:

動的書き換え可能な回路を搭載した半導体装置において、動的書き換え可能な回路の構成にかかわらず、ソフトウェアの互換性を維持する機構を提供する。

ソフトウェアが実行されると同時に、動的書き換え可能な回路を書き換えるデータ及びこれを動作させるドライバ・ソフトウェアを自動生成し、元のプログラムを途中から置き換える。

このように、ソフトウェアの互換性を維持することで、既存のソフトウェア資産を流用しつつ様々な機器で同じソフトウェアを用いることができる。

明 細 書

半導体装置

5 技術分野

本発明は、動的に書き換え可能な回路を搭載した半導体装置の構成と、その利用方法に関する。

背景技術

- 10 近年、情報処理機器の普及と高性能化に伴い、様々なアプリケーションが登場しており、これらアプリケーションはソフトウェアで記述され、汎用プロセッサで実行される形態が主流となっている。ところが、アプリケーションの中には汎用プロセッサよりも高度な演算処理能力を要求するものもあり、プロセッサにはより一層の処理能力の向上が要求されるようになっている。
- 15

- このため、汎用プロセッサに加えて特定のアプリケーションに特化した専用回路を1チップに搭載することにより、処理能力を向上させる例がある。さらに、この専用回路を動的に書き換え可能な回路(Dynamically Reconfigurable Circuit、以下、DRCと称する)により構成する例が、
- 20 特開平10-4345号公報や特開平10-335462号公報に開示されている。

- かかる先行技術では、DRC書き換えデータをアプリケーションのソフトウェア作成時にあらかじめ作成しておく。DRC書き換えデータによってDRCを書き換えることによって、DRCは特定のアプリケーションのための専用回路として機能するようになる。汎用プロセッサで実行されるソフトウェアには、DRC書き換えデータと書き換え命令とが
- 25

含まれている。

これにより、汎用プロセッサはアプリケーションの実行中にDRCを書き換えて専用回路として機能させることにより、その処理能力を向上させることができる。

- 5 本願の発明者らは、以上の先行技術のようにDRC書き換えデータとドライバ・ソフトウェアとを利用するDRCとチップ構成にあわせてあらかじめ用意し、DRC書き換え命令とDRC書き換えデータをソフトウェア内に記述する構成では、DRCの構成が異なるチップではそのソフトウェアの実行ができなくなってしまうという問題があることに気づ
- 10 いた。このことは、DRCの構成により適用できるソフトウェアの範囲が制限を受けることとなり、同じ命令セットを持つプロセッサソフトウェアでありながら、DRCの構成が異なるためにソフトウェアが利用できないという事態を生じさせることになる。

- 本発明の目的は、DRCを利用して処理能力を向上させながらも、D
- 15 RCの構成によらずソフトウェアの互換性を確保できる半導体集積回路装置を提供することである。

発明の開示

- 本願において開示される発明のうち代表的なものの概要を簡単に説明
- 20 すれば下記の通りである。

- 演算命令を含むソフトウェアを実行する半導体装置であって、複数の演算セルと複数のレジスタセルとを含み、演算セルが実行する演算種と複数の演算セル及び複数のレジスタセル間の配線接続とを設定可能な演算回路と、ソフトウェアに基づき、演算セルの演算種及び配線接続を設定
- 25 する設定データと、演算回路を用いてソフトウェアと等価な処理を行うためのドライバ・ソフトウェアとを生成する制御回路とを有する。

ここで演算種とは、セル演算の実行可能な論理和、論理積、排他的論理和といった論理演算、加減乗除といった算術演算、比較演算が含まれる。このような構成により、半導体装置上でドライバ・ソフトウェアを生成することが可能になり、ソフトウェアの互換性が確保できる。さらに、ソフトウェアの実行中にドライバ・ソフトウェアを作成することにより、ドライバ・ソフトウェアを作成するオーバーヘッドをユーザが意識することなく、演算回路を用いた高速処理が可能になる。

また、演算命令を含むソフトウェアを実行する半導体装置であって、レジスタと、演算器と、複数の演算セルと複数のレジスタセルとを含み、演算セルが実行する演算種と複数の演算セル及び複数のレジスタセル間の配線接続とを設定可能な演算回路と、ソフトウェアを格納する第1メモリ領域と、演算回路を用いてソフトウェアと等価な処理を行うためのドライバ・ソフトウェアを格納する第2メモリ領域と、実行するソフトウェアを制御する制御回路とを有し、ソフトウェアの処理は n 回繰り返され、第1回から第 i 回 ($i < n$) までの処理は、レジスタと演算器とを用いて第1メモリから読み出されたソフトウェアを実行することによって行われ、制御回路は第 i 回の処理を受けて、実行するソフトウェアをドライバ・ソフトウェアに切り換えることにより、第 $i + 1$ 回から第 n 回までの処理は、演算回路を用いて第2メモリ領域から読み出されたドライバ・ソフトウェアを実行することによって行われる。このようなソフトウェアとドライバ・ソフトウェアとを別のメモリ領域に記憶し、制御回路がソフトウェアとドライバ・ソフトウェアとを切り換える構成により、ソフトウェアの互換性が確保できる。

このような構成は、特に複数回繰り返されるソフトウェア（例えば、ループを形成しているソフトウェア）に有効であり、このようなループは画像処理や音声処理でよく現れる処理である。

図面の簡単な説明

第1図は、本発明の半導体集積回路装置の構成を示すブロック図である。

- 5 第2図は、DRCドライバSWの生成及び実行のタイミングを示すための図である。

第3図は、DRCの構成例を示すブロック図である。

第4図は、DRCの構成要素である入出力レジスタ・セルIORCの構成例を示すブロック図である。

- 10 第5図は、DRCの構成要素である演算セルCCの構成例を示すブロック図である。

第6図は、書き換え判定ユニットCDUの構成例を示すブロック図である。

第7図は、書き換え判定ユニットCDUの動作を示すフロー図である。

- 15 第8図(A)は、通常SW、第8図(B)は通常SWからHW/SW生成ユニットGUによって切り出された通常SW、第8図(C)はHW/SW生成ユニットGUによって作成されたDRCドライバSW、第8図(D)はプログラム中の命令の動作一覧である。

第9図は、HW/SW生成ユニットGUの動作を示すフロー図である。

- 20 第10図は、CDFGの例である。

第11図は、縮約処理を行ったCDFGの例である。

第12図は、ALAP (As Late As Possible) アルゴリズムを用いてスケジューリングを行ったCDFGの例である。

- 25 第13図は、さらにASAP (As Soon As Possible) アルゴリズムを用いてスケジューリングを行ったCDFGの例である。

第14図は、本発明の半導体集積回路装置の別の構成を示すブロック

図である。

第15図は、本発明の半導体集積回路装置のさらに別の構成を示すブロック図である。

5 発明を実施するための最良の形態

以下、本発明による代表的な実施の形態を図面に従って詳細に説明する。なお、以下においては、同じ参照符号は同じもの、もしくは類似のものを表す。

- 本発明では、DRCを用いることなく汎用プロセッサで実行されるアプリケーションソフトウェア（以下、このようなソフトウェアを「通常SW」と呼ぶ）から、アプリケーションソフトウェアの一部分をDRCで実行させるためのDRCドライバ・ソフトウェア（以下、このようなソフトウェアを「DRCドライバSW」と呼ぶ）を自動生成し、汎用プロセッサは、通常SWの一部をDRCドライバSWに置き換えて実行することで処理能力を向上させる。
- 10 アプリケーションソフトウェア（以下、このようなソフトウェアを「通常SW」と呼ぶ）から、アプリケーションソフトウェアの一部分をDRCで実行させるためのDRCドライバ・ソフトウェア（以下、このようなソフトウェアを「DRCドライバSW」と呼ぶ）を自動生成し、汎用プロセッサは、通常SWの一部をDRCドライバSWに置き換えて実行す
- 15 ることで処理能力を向上させる。

- 通常SWとDRCドライバSWとの関係を第8図（A）～（D）を用いて説明する。第8図（A）の例は高速フーリエ変換のプログラムの一部である。高速フーリエ変換はマルチメディア処理でよく用いられる処理である。第8図（A）のプログラムは汎用プロセッサの命令セットにより記述されている。プログラムの各命令の意味は第8図（D）に示している。第8図で用いた命令セットは一例であって、本発明が当該命令
- 20 より記述されている。プログラムの各命令の意味は第8図（D）に示している。第8図で用いた命令セットは一例であって、本発明が当該命令セットに限定されるものではないことは、以下の説明から明らかである。

- 本実施例では通常SWのうち繰り返し実行されるソフトウェア部分（ループ）を、DRCに実行させる。これは、DRCドライバSWを通常SWの実行中に自動生成するため、複数回実行されるソフトウェア部
- 25 （ループ）を、DRCに実行させる。これは、DRCドライバSWを通常SWの実行中に自動生成するため、複数回実行されるソフトウェア部

分をDRCに実行させるようにすることが効率的であると考えられるためである。第8図(A)の例では、第4行目から第18行目までのプログラムがその対象となる(第8図(B))。

本発明では、第8図(B)の通常SWに基づき、DRC書き換えデータとDRCドライバSW(第8図(C))とを生成する。DRC書き換えデータは、DRCが第8図(B)のプログラムで実行される演算を実行するようにDRCを設定するためのデータである。また、DRCドライバSWは、通常SWのプログラム中のDRCでは実行できない命令を実行し、汎用プロセッサからDRCへデータを入力し、DRCで実行された結果を汎用プロセッサに戻すためのプログラムである。したがって、汎用プロセッサは、汎用レジスタGRと演算器ALUを用いて第8図(B)を実行する代わりに、第8図(C)を実行してDRCを用いて演算を実行することにより、処理能力を向上させることができるのである。

第1図に本発明のLSIチップ100の構成を示す。LSIチップ100は、バス・ステート・コントローラBSCと、ダイレクト・メモリ・アクセス・コントローラDMACと、オンチップ・メモリOCMと、DRCドライバSW格納メモリDSMと、命令キャッシュICHと、データ・キャッシュDCHと、キャッシュ制御ユニットCCNと、CPU101により構成される。CPU101はDRC制御ユニット102と、命令バッファIBFと、命令フェッチ・ユニットIFUと、セクタSELと、命令デコーダIDCと、汎用レジスタGRと、演算器ALUと、DRCにより構成される。DRC制御ユニット102は、書き換え判定ユニットCDUと、ハードウェア/ソフトウェア生成ユニットGU(以下、HW/SW生成ユニットと記述する)により構成される。

CPU101が実行する命令は命令キャッシュICHに格納され、命令フェッチ・ユニットIFUの命令ロード信号に従って命令キャッシュ

I C H内に格納されている命令は命令バッファ I B Fに転送される。同時に、書き換え判定ユニット C D Uは、命令キャッシュ I C Hから命令バッファ I B Fへ転送される命令を常に監視する。

- 書き換え判定ユニット C D Uは第 8 図 (A) の例でいえば、条件分岐
- 5 命令 B Fを検出し、それに基づき D R Cで実行する候補となるプログラムとして第 8 図 (B) に示すソフトウェア部分を切り出し、格納する。D R Cで実行するソフトウェア部分が決定すると、書き換え判定ユニット C D Uは H W / S W生成ユニット G Uに対して、D R C書き換えデータの作成、D R C書き換え、及び D R Cドライバ S Wの作成を指示する。
- 10 H W / S W生成ユニット G Uは、切り出されたプログラムから D R C書き換えデータを生成し、D R Cの書き換えを行う。また、書き換えた D R Cを利用するための D R Cドライバ S Wを生成し、生成された D R Cドライバ S Wはドライバ S W格納メモリ D S Mに格納する。これらの処理が終了すると、H W / S W生成ユニット G Uは書き換え判定ユニット C D Uに終了を通知するとともに、D R Cドライバ S Wが格納される先頭アドレスを通知する。
- 15

なお、通常 S Wはオンチップ・メモリ O C Mや外部メモリ・チップ E X T Mに格納されているのが一般的である。

- C P U 1 0 1によるプログラムの実行は以下ようになる。第 8 図の
- 20 例により説明する。最初は第 8 図 (A) の通常 S Wがそのまま実行される。H W / S W生成ユニット G Uから終了通知を受けた後に、書き換え判定ユニット C D Uは、命令キャッシュ I C Hから命令バッファ I B Fへ転送される通常 S Wの実行中の命令が、D R Cドライバ S Wに置き換えられるプログラムの条件分岐命令 (第 8 図 (B) の第 1 5 行、すなわち第 8 図 (A) の第 1 8 行に相当する) まで処理が進んだことを確認すると、命令フェッチ・ユニット I F Uを介してセクタ S E Lを切り替
- 25

え、命令デコーダIDCに入力される命令を命令バッファIBFから判定ユニットCDUに切り替える。次に、条件分岐命令の分岐先アドレスをDRCドライバSWの格納される先頭アドレスに変換した命令を、セレクトSELを介して命令デコーダIDCに出力する。次のCPUクロック・サイクルでは、書き換え判定ユニットCDUは命令フェッチ・ユニットIFUを介してセレクトSELを切り替え、命令デコーダIDCに入力される命令を判定ユニットCDUから命令バッファIBFに切り替える。これにより、以降はDRCドライバSWが実行される。DRCドライバSWの最後の命令は、切り出した通常SWの次の命令が格納されたアドレスへの無条件分岐命令（第8図（C）第15行）であり、この命令の実行によって通常SWの実行に戻る。

ここで、命令キャッシュICHへの命令の格納は、命令デコーダIDCの実行を受けて、キャッシュ・コントローラCCNによって行われる。キャッシュ・コントローラCCNは、プロセッサ・バスPRCB上のモジュールであるオンチップ・メモリOCMと、ダイレクト・メモリ・アクセス・コントローラDMACと、バス・コントローラBSC（外部メモリ・チップEXTMへのアクセスの場合）に加えてドライバSW格納メモリDSMに対してもメモリ・アクセスを実行可能に構成されている点の一つの特徴である。

第2図にDRCドライバSWの生成及び実行のタイミングを示す。

まず通常SWが実行され、命令フェッチ・ユニットIFU内のプログラム・カウンタに格納されている現在実行中の命令のアドレスより前のアドレスへの条件分岐が起こった場合（第8図（A）第18行）には、書き換え判定ユニットCDUは、通常SW中にループがあると仮決定する。

書き換え判定ユニットCDUは、以降の実行で命令キャッシュICH

から命令バッファ I B F へロードされる命令、すなわち第 8 図 (A) 第 4 行～第 1 8 行の命令を取得して格納する。再度、条件分岐命令により格納された先頭の命令に戻った場合には、格納された命令はループを形成していると判定し、D R C の利用を本決定する。

- 5 H W / S W 生成ユニット G U は、3 回目のループで D R C 書き換えデータの作成、D R C ドライバ S W の作成、および D R C の書き換えを行う。3 回目のループにおいて D R C の利用が可能になるので、C P U 1 0 1 は、演算ユニット A L U による演算に代えて、第 4 回目以降のループは D R C ドライバ S W を実行することにより D R C により演算を行う
- 10 ようにする。なお、D R C の書き換えが 3 回目のループ実行中に終わらない場合は、C P U 1 0 1 は D R C の書き換えが終了する時点を含むループの最後まで通常 S W を実行することになる。

- 次に、D R C の構造を第 3 図～第 5 図を用いて詳しく説明する。第 3 図は D R C の内部構成を示している。入出力レジスタセル I O R C と、
- 15 演算セル C C と、入出力レジスタセル I O R C へのデータ入力ポート 2 0 0 と、レジスタ指定入力ポート 2 0 1 と、セル入力線 2 0 3 a、2 0 3 b、2 0 3 c と、D R C のデータ出力ポート 2 0 2 と、各セルからのセル出力線 2 0 4 a、2 0 4 b と、配線領域 2 0 5 と、配線プログラム素子 2 0 6 から構成される。配線プログラム素子 2 0 6 は、配線領域 2
- 20 0 5 中の配線同士の結線を決定するためのスイッチ素子とこれらスイッチ素子のオン／オフ状態を記憶するための図示しない記憶素子（例えば、S R A M、F l a s h メモリ等）により構成される。

- D R C へのデータ入力時には、命令デコーダ I D C からレジスタ指定入力ポート 2 0 1 にレジスタ指定信号が入力され、1 つの入出力レジスタ
- 25 タ・セル I O R C が選択される。データはデータ入力ポート 2 0 0 より入力され、選択された入出力レジスタ・セル I O R C のみに入力される。

また、DRCからのデータの出力時には、命令デコーダIDCからレジスタ指定入力ポート201にレジスタ指定信号が入力される。これにより出力セクタOSELが切り換えられ、1つの入出力レジスタ・セルIORCの出力が選択される。データは、セル出力線204aより出力
5 セクタOSELに入り、選択されたデータのみデータ出力ポート202より出力される。

第4図は、第3図の入出力レジスタ・セルIORCの構成を示している。入出力レジスタ・セルIORCは入力セクタISELとセル・レジスタCRにより構成されている。入力セクタISELは、レジスタ
10 指定入力ポート201からの入力に応じて、配線からのセル入力線203aまたはデータ入力ポート200からの入力に切り換える。入力セクタISELを介して入力されたデータは、セル・レジスタCRに保持される。このセル・レジスタCRはクロック入力301と同期して動作し、リセット入力302によってリセットされる。なお、第3図ではク
15 ロック及びリセットの配線は省略しているが、全ての入出力レジスタ・セルIORCと演算セルCCとに接続されている。セル・レジスタCRに保持されたデータはセル出力線204aより外部に出力される。

なお、本実施例ではデータは8ビット単位で入出力が行われているが、この大きさには限定されない。

20 第5図は、第3図の演算セルCCの構成を示している。演算セルCCは、セル演算ユニットCALUと、フリップ・フロップFFと、演算プログラム素子400から構成される。セル演算ユニットCALUは、CPU内のALUと同じ機能を持ち、演算プログラム素子400によって
25 どの演算機能を使用するかを設定する。セル演算ユニットCALUの実行可能な論理和、論理積、排他的論理和といった論理演算、加減乗除といった算術演算、比較演算のうちから、演算プログラム素子400はセ

ル演算ユニットCALUが実行する演算を設定する。

このように、演算セルCCの演算プログラム素子400の設定により、演算セルCCの演算内容を決定できる。また、配線プログラム素子206の設定により、いかなるデータを入出力レジスタ・セルIORC、演算セルCCに入力するのか、また入出力レジスタ・セルIORCのデータまたは演算セルCCの演算結果をどこに出力するのかを設定することができる。このようにDRC書き換えデータは、演算プログラム素子400の設定値と配線プログラム素子206の設定値とを含み、所望の演算を実行可能にする。

- 10 第6図に書き換え判定ユニットCDUの構成を、第7図に書き換え判定ユニットCDUの動作を示す。書き換え判定ユニットCDUは、分岐アドレス格納バッファBABと、ループ・カウンタLCと、命令アドレス判定ユニットIADUと、DRC状態レジスタDSRと、通常SW一時格納バッファTBFと、分岐コントローラBCLから構成される。さらに、DRC状態レジスタDSRは、HW/SW生成ユニットGUの状態を示す部分、DRCドライバSWへの分岐アドレスを格納する部分、およびDRCドライバSWに置き換える通常SWの次の命令があるアドレスを格納する部分の3つの部分で構成される。

以下、第7図に即して書き換え判定ユニットCDUの動作を説明する。

- 20 まず、命令キャッシュICHから命令バッファIBFへ送られる命令を命令アドレス判定ユニットIADUに取り込む(500)。命令アドレス判定ユニットIADUは、命令フェッチ・ユニットIFU中のプログラム・カウンタPC内のアドレスを取り出し、その現在実行中の命令のアドレスがドライバSW格納メモリDSMのアドレス領域に該当するかどうかを判定する(501)。

該当する場合は現在DRCドライバSWを実行中であることを意味す

る。このとき、ループカウンタLCが0以外の値であれば(502) DRCドライバSWの第1行の命令(即ち第8図(C)の第1行のMOV命令)であるため、ループ・カウンタLCを0にリセットし、分岐コントローラBCLは、セクタSELに命令バッファIBF側に切り換えさせる。ループカウンタLCが0の場合には何もしない。ループ・カウンタLCには、通常SW中で連続して実行されるループの回数が保持されており、リセット信号によりリセットされる。

該当しない場合には現在通常SWを実行中であることを意味するので、命令アドレス判定ユニットIADUは、命令が条件分岐命令であるかどうかを判定する。条件分岐命令ではなく、かつこのときループ・カウンタLCの値が1であれば、第2回目のループを実行していることになる。そのため、通常SWを取得するため(第2図を参照)、命令アドレス判定ユニットIADUは命令を通常SW一時格納バッファTBFBに格納する。

15 ステップ505において、命令アドレス判定ユニットIADUは、DRC状態レジスタDSRをチェックする。DRC状態レジスタDSRは、DRCの状態を示す第1レジスタと、DRCドライバSWへの分岐アドレス(例えば第8図(C)の第1行のMOV命令が格納されたアドレス)を格納する第2レジスタと、DRCドライバSWに切り換えられる通常SWの次のアドレス(例えば第8図(A)のL003:(第19行))を格納する第3アドレスとから構成される。第1レジスタには「DRC利用不可」、「DRC利用準備中」、「DRC利用準備完了」の3状態のいずれかが格納されている。CPUリセットを受けて「DRC利用不可」の値に更新され、HW/SW生成ユニットGUによるDRC利用準備開始時に「DRC利用準備中」の値に更新され、DRC利用準備完了時に「DRC利用準備完了」の値に更新される。DRC利用準備完了時に第

2 アドレスは更新され、GUへの動作開始の通知時に第3アドレスは更新される。

DRC状態レジスタDSRの第1レジスタが「DRC利用準備完了」の値であれば、分岐コントローラBCLは、命令フェッチ・ユニットIFUを介してセクタSELをDRC制御ユニットCDUに切り換えて、書き換え判定ユニットCDUからの出力が命令デコーダIDCに繋がるようにする。その後、分岐命令の分岐先アドレスをドライバSW格納メモリDSM内のDRCドライバSWの先頭アドレスに変更した分岐命令を送る。

- 10 DRC状態レジスタDSRの第1レジスタが「DRC利用準備完了」の値以外であれば、ループの存在の判定を行う（第2図における仮決定処理）。まず、命令アドレス判定ユニットIADUは、現在のプログラム・カウンタPCと分岐先アドレスとを比較する（513）。分岐先アドレスの方が大きい場合はループは存在しないので、ループ・カウンタLCを0とする（514）。プログラム・カウンタPCの方が大きい場合には、さらに命令アドレス判定ユニットIADUは、分岐アドレス・バッファBABに格納された分岐先アドレスと、分岐命令の分岐先とを比較する。分岐アドレス・バッファBABは、分岐命令を実行した場合にその分岐先アドレスを上書き保存するバッファである。したがって、
- 20 分岐先が分岐アドレス・バッファBABのアドレスに等しければ、ループの存在を決定し、DRC書き換えデータ、DRCドライバSWの生成処理に移る。具体的には、ループ・カウンタに1を加算し（509）、その値が2であれば（510）、命令アドレス判定ユニットIADUは、分岐命令を通常SW一時格納バッファTBFに格納し、DRC状態レジスタDSRの第3レジスタに現在のPCの値を1つ進めた値（例えば図
- 25 8（A）のL003：（第19行）のアドレス）を代入して、HW/S

W生成ユニットGUに対してDRC利用準備の開始信号を送る(511)。

一方、分岐先が分岐アドレス・バッファBABのアドレスと異なっていれば、新たなループの存在の可能性がある。そこで、命令アドレス判定ユニットIADUは、ループ・カウンタLCに1を代入し、通常SW

5 一時格納バッファTBFを一括クリアする。

上記動作を第8図のプログラム例に基づいて説明すると以下のようになる。

第1回目のループの実行では、第8図(A)の第1行～第17行の命令は、ループ・カウンタLCは0であり、第7図のフローでは何も実行
10 されない。18行目の条件分岐命令BFが取り込まれると、ループ・カウンタLCを1にするとともに(508)、分岐先アドレス(第4行)が分岐アドレス・バッファBABに上書きされる。

第2回目のループの実行では、第5行～第17行目の各命令は、ループ・カウンタLCが1なので、通常SW一時格納バッファTBFに格納
15 される(504)。第18行目の条件分岐命令BFが取り込まれると、その分岐先アドレスと分岐アドレス・バッファBABに格納されたアドレスとは一致するので(507)、ループ・カウンタLCを2として(509)、条件分岐命令BFを通常SW一時格納バッファTBFに送るとともに、HW/SW生成ユニットGUにDRC利用準備を開始させる(5
20 10, 511)。

第3回目のループの実行では、第5行～第17行目の各命令は、ループ・カウンタLCは2であるので、第7図のフローでは何も実行されない。第18行目の条件分岐命令BFが取り込まれ、DRCが利用可能な状態になっていれば、分岐命令の分岐先アドレスをDRCドライバSW
25 の先頭アドレス(第8図(C)第1行)に変更することで、以降のループの実行はDRCを利用して行われる。

DRCドライバSWの実行では、JMP命令（第8図（C）の第15行）により通常SWのL003（第8図（A）の第19行）に分岐し、通常SWの実行に復帰する。

次に、HW/SW生成ユニットGUの動作を第9図のフロー図を元に
5 説明する。HW/SW生成ユニットGUは、書き換え判定ユニットCDUからの命令を受け、通常SW一時格納バッファTBFから通常SW（第8図（B））を取得するとともに、DRC状態レジスタDSRの第1レジスタには「DRC利用準備中」の値を入力する（600）。

HW/SW生成ユニットGUは、まず取得した通常SWから第10図
10 に示すようなCDFG (Control Data Flow Graph)を作成する（601）。CDFGは、入力された通常SW中の各命令をノード（命令）で、命令オペランドのデータ依存関係をエッジ（矢印）で表したものである。なお、第10図におけるブロックの括弧内の数字は第8図（B）の対応する行数を表している。

15 第10図は、第8図（B）の通常SWからそのままCDFGを作成したものである。依存関係には、「制御上の依存関係」と「データ上の依存関係」の2種類がある。「制御上の依存関係」とは、例えばDT命令でレジスタR0の値を設定し、レジスタR0の値に応じて条件分岐命令BFを実行するといった依存関係である。これは汎用プロセッサの命令
20 セットに依存するものであり、あらかじめ依存関係を登録しておく必要がある。「データ上の依存関係」は、例えばMOV命令で転送されたデータを用いてSUBで演算を行うといった依存関係である。したがって、プログラムの内容によって決定していく必要がある。オペランドに他の命令のオペランドと依存関係のないものを最上位に配置し、依存関係に
25 応じてより下位に配置する。「データ上の依存関係」は具体的には以下のようにして決定する。

第8図(B)第2行~第4行、第10行の命令のオペランドはそれより前の命令のオペランドに依存関係がないので、同列に位置する。「MOV @R6, R2」とは「レジスタR6の指示するアドレスに格納されたデータをレジスタR2に転送する」という意味である。すなわち、オペランドの
5 右の値が@を含むMOV命令は外部からレジスタにデータを読み込むという命令であり、さらに@のついたオペランドは第8図(B)内で初の出現であるため、「データ上の依存関係」としては最上位に位置することになる。第5行の命令「MUL」は、レジスタR2, R3のデータを用いるため、第3行~第4行の命令と依存関係を有する。第7行の命令は
10 「MUL」の実行結果はレジスタMACLに入力されるため、レジスタMACLのデータをレジスタR7に転送するためのものである。

以下のプログラムについても同様にして依存関係を決定するが、エッジの決定には同じレジスタ名であっても内容が更新されている場合があることに注意する必要がある。例えば、第8行の「SUB R7, R1」は「レジスタR7のデータとレジスタR1のデータの差をレジスタR1に格納する」という意味である。第8行の命令の実行によりデータが更新されているため、第9行の「MOV R1, @R5」は第8行の命令と依存関係はあるが、
15 第6行の「MOV @R4, R1」とは依存関係が存在しない。

このように、着目する命令に対して、それ以前に実行された命令のうち、オペランドに共通のレジスタ名を有する命令と依存関係を設定することでCDFGが作成できる。ただし、共通のレジスタ名を有する命令が複数ある場合にはレジスタのデータが更新される可能性を考慮して、
20 直前に実行された命令との間に依存関係を設定する。

第10図のCDFGを縮約したのが、第11図のCDFGである。縮約することでDRCの回路構成を単純にし、DRCでの演算処理が高速に行えるといった効果がある。
25

- 第1の縮約の形態が701の2命令(第10図)を801の1命令(第11図)に縮約するものである。STS命令は一切の演算処理を行うことなくデータをレジスタ間で転送する命令である。このような転送は汎用プロセッサで処理する場合には命令セットの構成上必要となる命令ではあるものの、DRCで処理する場合には直接必要なデータの格納されたレジスタと演算セルとを接続するようにすればよいため、このような命令をDRCの構成に反映させる必要はない。このような、データをレジスタ間で転送する命令はあらかじめ縮約対象とする命令として事前に登録しておくことで第1の縮約を実行できる。
- 10 第2の縮約の形態が702の4命令(第10図)を802の3命令(第11図)に縮約するものである。汎用プロセッサでは時系列で処理がされるため、レジスタ名が違っていてもその内容が同じであるということが生じうる。DRCで処理する場合には必要なデータの格納されたレジスタと演算セルとを接続するようにすればよいため、このようなレジスタ名の相違をDRCの構成に反映させる必要はない。そのため、演算命令(第8図の例ではMUL命令、ADD命令、SUB命令)については、そのレジスタのデータの内容を比較し、レジスタが異なっても同じデータが格納されている場合には縮約するようにする。例えば、第8行のSUB命令と第11行のADD命令のオペランドは、レジスタR7が
- 15 共通し、レジスタR1、レジスタR3とで異なっている。しかしながら、それぞれデータ上の依存関係をたどるといずれのレジスタに格納された内容も「レジスタR4の指示するアドレスに格納されたデータ」であるため、上記のような縮約が可能になる。

- 次に、HW/SW生成ユニットGUは、ステップ601で作成した第
- 25 11図に示すCDFGの各ノードに対して、スケジューリング、すなわちハードウェア・リソースの制約を考慮した上で各ノードを実行するク

ロック・サイクルの割り当てを行う（602）。

このスケジューリングの一方法としてALAPアルゴリズムとASAPアルゴリズムを第11図のCDFGに対して併用して適用した例を第12図及び第13図に示す。ALAP（As Late As Possible）アルゴリズムは同時に実行可能な命令を後から詰めて行くスケジューリングである。同時に実行できない命令の第一は相互に依存関係のある命令である。第二は、DRCの外部にアクセスする命令はそのサイクルに1つしか実行することができない。この結果が第12図である。

さらに、第12図のスケジューリングに対して、同時に実行可能な命令を前から詰めて行くASAP（As Soon As Possible）アルゴリズムを適用したものが第13図である。できるだけ、命令を実行するサイクルを前に詰めることによって、処理時間に余裕が生まれる。

1サイクルに含まれる命令は1クロック・サイクルで実行することができ、サイクル1～8の8クロック・サイクルで全ての命令を実行することができる。

HW/SW生成ユニットGUは、第13図に示すスケジューリングされたCDFGより、DRC書き換えデータを生成する。具体的には、以下の通りである。ノード部分を入出力レジスタ・セルIORC及び演算セルCCに割り当てる。MOV命令については入出力レジスタ・セルIORCを割り当て、演算命令については演算セルCCを割り当てる。演算命令の内容に応じて、演算セルの演算内容の設定を行う演算プログラム素子400の設定データを作成する。また、CDFGのエッジはデータの流れを示しているので、CDFGのエッジの接続関係にしたがって、入出力レジスタ・セルIORC及び演算セルCCが相互に配線されるように、配線プログラム素子206の設定データを作成する（603）。この演算プログラム素子400の設定データ及び配線プログラム素子2

06の設定データがDRC書き換えデータである。

HW/SW生成ユニットGUは、DRC書き換えデータにしたがってDRCの演算プログラム素子303と配線プログラム素子206を書き換える(604)。これと並行してDRCドライバSWの作成を行う(605)。

DRCドライバSWの作成方法について説明する。第13図のCDFGから第8図(C)に示すDRCドライバSWを生成する。

まず、汎用レジスタGRに格納されているデータをDRCの入出力レジスタ・セルIORCに移動させる必要がある。例えば、第8図(B)第3行の「MOV @R6, R3」という命令は「汎用レジスタR6の指示するアドレスに格納されたデータを汎用レジスタR3に転送する」命令である。この命令をDRCで実行するためには、まず汎用レジスタR6のデータをDRC中に移動させておく必要がある。したがって、第8図(C)第1行「MOV R6, dR6」という命令が設けられる。この命令は「汎用レジスタR6に格納されたデータ(アドレス)をDRCの入出力レジスタ・セルdR6に転送する」という意味である。このような命令が第8図(C)の第1行～第3行に設けられている。

その後、第13図の各ノードのうち、DRC以外で実行される命令がDRCドライバSWに設けられる。第8図(C)の第5行～第12行の命令はそれぞれ第13図のサイクル1～8に含まれるノードに対応している。

第8図(C)第5行「MOV @dR6, dR3」は「入出力レジスタ・セルdR6の指示するアドレスに格納されたデータを入出力レジスタ・セルdR3に転送する」命令であり、これは第13図のサイクル1の「MOV @R6, R3」に対応している。以下同様であるが、例えばサイクル3では「MOV @R4, R1」、「MUL」の2つのノードが存在しており、後者はDRCが実行する

命令であるから、DRCドライバSWには現れない。また、サイクル6は「ADD」というDRCが実行する命令しか存在していない。その場合には、空きサイクルが発生しているとして、ノー・オペレーション命令(NOP)を追加する。

- 5 なお、演算命令についてはDRCで実行させるために、演算に必要なデータは汎用レジスタGRではなく、DRCの入出力レジスタ・セルIO RCに格納することになる。そのため、演算実行後の値を汎用レジスタGRに戻す必要がある。したがって、第8図(C)の第13行と第14行の命令が設けられている。例えば、「MOV dR5, R5」は「入出力レジスタ・セルdR5に格納されたデータを汎用レジスタR5に転送する」
10 命令である。

- 最後に、通常SWの実行に復帰するために、通常SWにおける条件分岐命令の次のアドレスへの無条件分岐命令(JMP)(第8図(C)第15行)を設ける。この無条件分岐命令の分岐先はDRC状態レジスタ
15 の第3レジスタに格納されたアドレスとする。

- このように、第13図のスケジューリングされたCDFGを元に第8図(C)のDRCドライバSWが生成できる。キャッシュのミスや割り込みなどの外的要因を排除して考えると、もとの通常SWが一回のループの実行に14サイクル必要だったのに対して、ループのコア部分(第
20 8図(C)の第5行~第12行)の8サイクルで実行できる。

- 作成されたDRCドライバSWは、ドライバSW格納メモリDSMに格納される。格納する場所は、初回ならばドライバSW格納メモリDSMの先頭アドレスとし、以降は前回書き込んだ部分の後ろに書き込む。書き込む領域が十分でない場合には、再びドライバSW格納メモリDS
25 Mの先頭アドレスから書き込むようにする。

ステップ605まで終了すると、HW/SW生成ユニットGUはDR

C状態レジスタDSRの第1レジスタに「DRC利用準備完了」の値を書き込み、DRC状態レジスタDSRの第2レジスタにステップ605で作成したDRCドライバSWの格納されたメモリの先頭アドレスを書き込む(606)。

- 5 以上に述べたフローにしたがってHW/SW生成ユニットGUを動作させることにより、プログラム実行時に自動的にDRC書き換えデータ作成および書き換えと、DRCドライバSWの作成を行うことができる。

- 第14図を用いて第1図の構成の変形例を説明する。この実施例は、第1図に示した第1の実施例のDRC、書き換え判定ユニットCDU、
10 HW/SW生成ユニットGUを、CPU101から分離した例である。具体的には、DRCをプロセッサ・バスPRCBに接続し、セクタSELの制御を書き換え判定ユニットCDUが直接制御する。また、命令バッファIBFをセクタSELの後に配置している。

- 第1図の構成との動作の違いは、DRCの入出力レジスタ・セルIORCへのアクセスがアドレスを指定して行われることである。例えば、
15 第8図(C)のDRCドライバSWの第1行目[MOV R6 dR6]などのようなDRCの入出力レジスタ・セルIORCにアクセスしている命令は、DRCが直接プロセッサ・バスPRCBに接続しているために、入出力レジスタ・セルIORCのアドレスが直接指定ではなく、間接指定となる。
20 例えば、[MOV R6 @R12]のようなアドレス指定になる。この命令は、汎用レジスタR6のデータを汎用レジスタR12で指示される入出力レジスタ・セルIORCに転送することを意味している。

- この変形例では、DRCをCPUの外部に配置することで、第1図の構成に比べてDRCのサイズを大きくすることが可能である。また、DRCと、書き換え判定ユニットCDUと、HW/SW生成ユニットGU
25 がCPU101と分離されているために、設計の変更が容易である。さ

らに、CPU以外のモジュールがDRCを利用することも可能である。
例えば、ダイレクト・メモリ・アクセス・コントローラDMACにより、
DRC上の入出力レジスタセルIORCにアクセスしてDRCを利用す
ることができる。この場合には、DRCとCPU101とで異なる処理
5 を同時に実行させることもできる。

第15図を用いて第1図の構成のさらに別の変形例を説明する。この
構成例では、第1図に示したDRC制御ユニット102の機能を、DRC
制御専用プロセッサDCPで実現している点異なる。本構成例では、
DRC制御ユニット102の機能をDRC制御専用プロセッサDCPで
10 実装するため、その内部の書き換え判定ユニットCDU、およびHW/
SW生成ユニットGUの動作をソフトウェアで実現する。従って、DRC
制御専用プロセッサDCPのソフトウェアを入れ替えるだけで容易に
DRC制御ユニット102を更新することができ、本技術を利用したLSI
チップ製造後においても本LSIの性能を向上させることが可能と
15 なる。もちろん、第14図の構成と第15図の構成を組み合わせで実施
することも可能である。

さらに、本発明は種々の変形が可能である。例えば、実行する通常SW
を予備的に実行しておき、DRC書き換えデータ及びDRCドライバ
SWを予め登録しておくようにしてもよい。この場合には通常SWの実
20 行中にDRC書き換えデータ及びDRCドライバSWの生成処理が不要
になるため、第2回目のループからDRCを用いた演算処理が行える。

本発明の構成により、プロセッサとDRCを搭載する半導体装置にお
いて、ソフトウェアからDRC書き換えデータとDRCドライバSWと
を自動生成することが可能になる。かかる構成により、DRCを利用す
25 る場合にもDRCにあわせた専用のプログラムを記述する必要がなくなり、
ソフトウェアの互換性を保つことができる。このように、ソフトウ

ウェアの互換性を維持することで、既存のソフトウェア資産を流用しつつ少なくとも同じ命令セットで機能するプロセッサであれば同じソフトウェアを用いることができる。

5 産業の利用可能性

本発明は、特に複数回繰り返されるソフトウェア（例えば、ループを形成しているソフトウェア）に有効であり、このようなループは画像処理や音声処理でよく現れる処理である。

請 求 の 範 囲

1. 演算命令を含むソフトウェアを実行する半導体装置であって、
レジスタと、
演算器と、

- 5 複数の演算セルと複数のレジスタセルとを含み、上記演算セルが実行する演算種と上記複数の演算セル及び上記複数のレジスタセル間の配線接続とを設定可能な演算回路と、

上記ソフトウェアに基づき、上記演算セルの上記演算種及び上記配線接続を設定する設定データと、上記演算回路を用いて上記ソフトウェア
10 と等価な処理を行うためのドライバ・ソフトウェアとを生成する制御回路とを有する半導体装置。

2. 請求項1において、

- 上記制御回路は、上記レジスタと上記演算器とを用いて上記ソフトウェアを実行している期間中に、上記設定データ及び上記ドライバ・ソフ
15 トウェアを生成するように構成された半導体装置。

3. 請求項2において、

上記ソフトウェアの処理は n 回繰り返され、

第1回から第 i 回 ($i < n$) までの処理は、上記レジスタと上記演算器とを用いて上記ソフトウェアを実行することによって行われ、

- 20 第 $i + 1$ 回から第 n 回までの処理は、上記演算回路を用いて上記ドライバ・ソフトウェアを実行することによって行われる半導体装置。

4. 請求項1において、

上記ドライバ・ソフトウェアは少なくとも上記レジスタから上記演算回路の上記レジスタセルへのデータ転送命令と、上記演算回路の上記レ

- 25 ジスタセルから上記レジスタへのデータ転送命令とを含む半導体装置。

5. 請求項1において、

制御回路は、上記設定データ及び上記ドライバ・ソフトウェアを生成するソフトウェアを実行することにより、上記設定データ及び上記ドライバ・ソフトウェアを生成する半導体装置。

6. 請求項 1 において、

5 上記演算回路はバスに接続された半導体装置。

7. 請求項 1 において、

上記ソフトウェアを実行するのに要するクロックサイクル数は上記ドライバ・ソフトウェアを実行するのに要するクロックサイクル数よりも小さい半導体装置。

10 8. 演算命令を含むソフトウェアを実行する半導体装置であって、

レジスタと、

演算器と、

複数の演算セルと複数のレジスタセルとを含み、上記演算セルが実行する演算種と上記複数の演算セル及び上記複数のレジスタセル間の配線

15 接続とを設定可能な演算回路と、

上記ソフトウェアを格納する第 1 メモリ領域と、

上記演算回路を用いて上記ソフトウェアと等価な処理を行うためのドライバ・ソフトウェアを格納する第 2 メモリ領域と、

実行するソフトウェアを制御する制御回路とを有し、

20 上記ソフトウェアの処理は n 回繰り返され、

第 1 回から第 i 回 ($i < n$) までの処理は、上記レジスタと上記演算器とを用いて上記第 1 メモリ領域から読み出された上記ソフトウェアを実行することによって行われ、

上記制御回路は上記第 i 回の処理を受けて、実行するソフトウェアを上記ドライバ・ソフトウェアに切り換えることにより、第 $i + 1$ 回から
25 第 n 回までの処理は、上記演算回路を用いて上記第 2 メモリ領域から読

み出された上記ドライバ・ソフトウェアを実行することによって行われる半導体装置。

9. 請求項8において、

上記制御回路は、上記ソフトウェアに基づき、上記演算セルの上記演
5 算種及び上記配線接続を設定する設定データと、上記演算回路を用いて
上記ソフトウェアと等価な処理を行うためのドライバ・ソフトウェアと
を生成する半導体装置。

10. 請求項8において、

上記制御回路は、演算回路の上記演算種及び上記配線接続を設定する
10 設定データを有し、

上記ソフトウェアを実行している期間中に、上記演算回路は、上記設
定データにより、上記演算セルの上記演算種及び上記配線接続を設定す
る半導体装置。

11. 請求項8において、

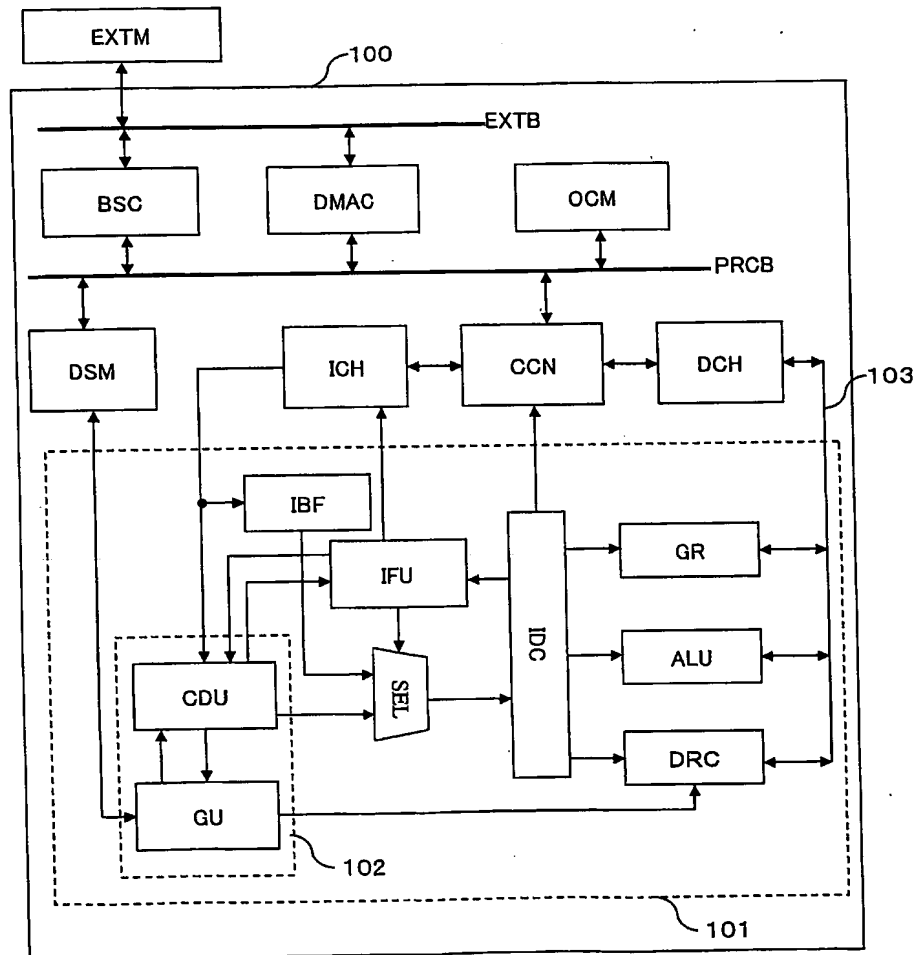
15 上記ドライバ・ソフトウェアは少なくとも上記レジスタから上記演算
回路の上記レジスタセルへのデータ転送命令と、上記演算回路の上記レ
ジスタセルから上記レジスタへのデータ転送命令とを含む半導体装置。

12. 請求項8において、

上記ドライバ・ソフトウェアを実行するのに要するクロックサイクル
20 数は上記ソフトウェアを実行するのに要するクロックサイクル数よりも
小さい半導体装置。

1/12

第1図



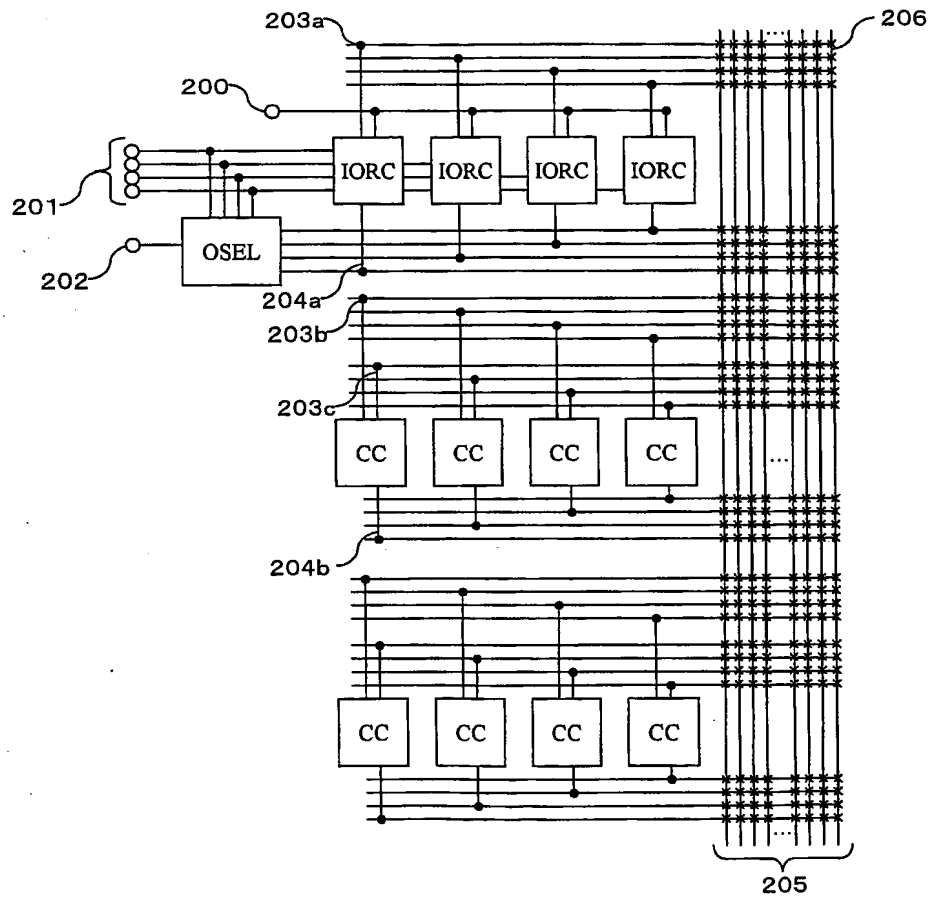
2/12

第2図

ループ 回数	DRC制御ユニット 102の動作	実行される ソフトウェア
1回目	仮決定	通常SW
2回目	通常SW取得 本決定	通常SW
3回目	DRC書き換え	通常SW
4回目		DRCDライブSW
5～n回目		DRCDライブSW

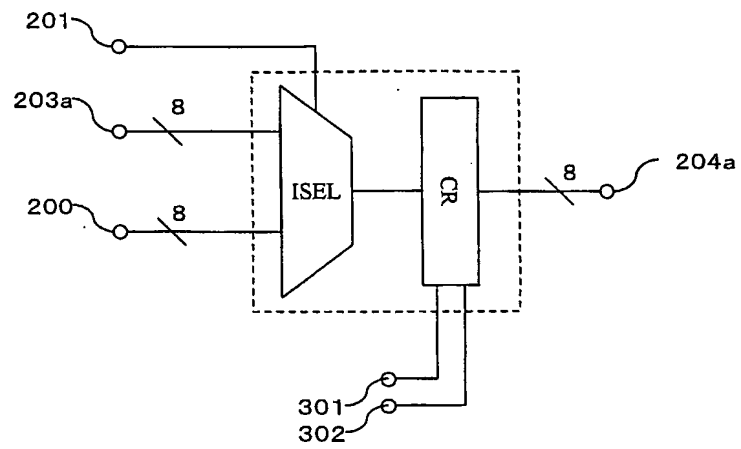
3/12

第3図

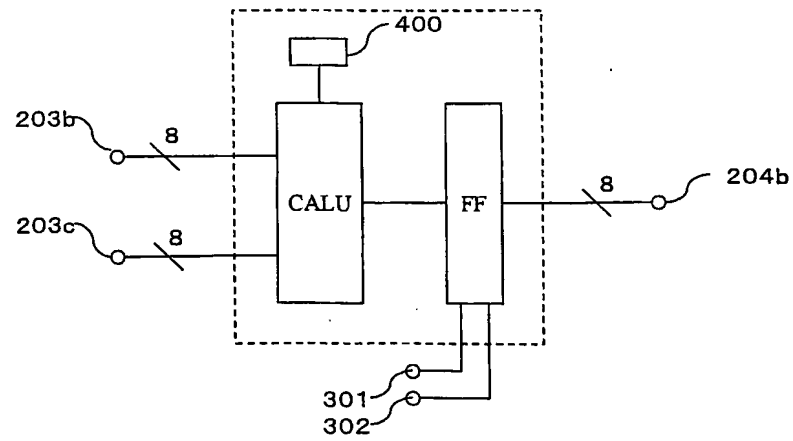


4/12

第4図

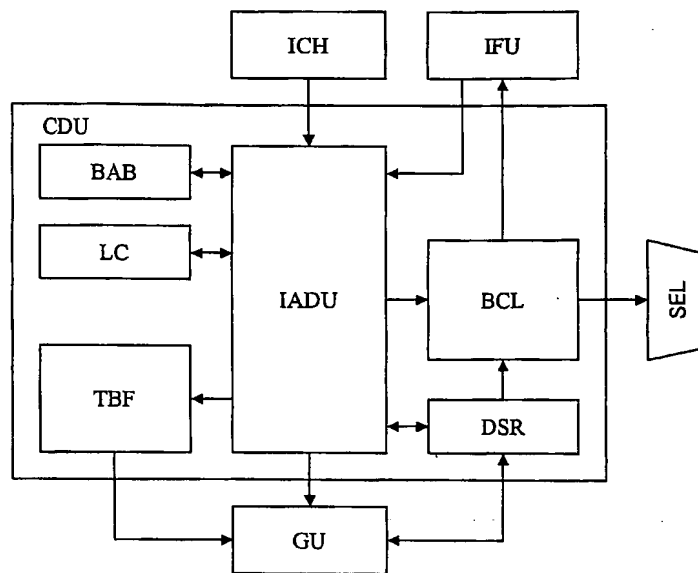


第5図



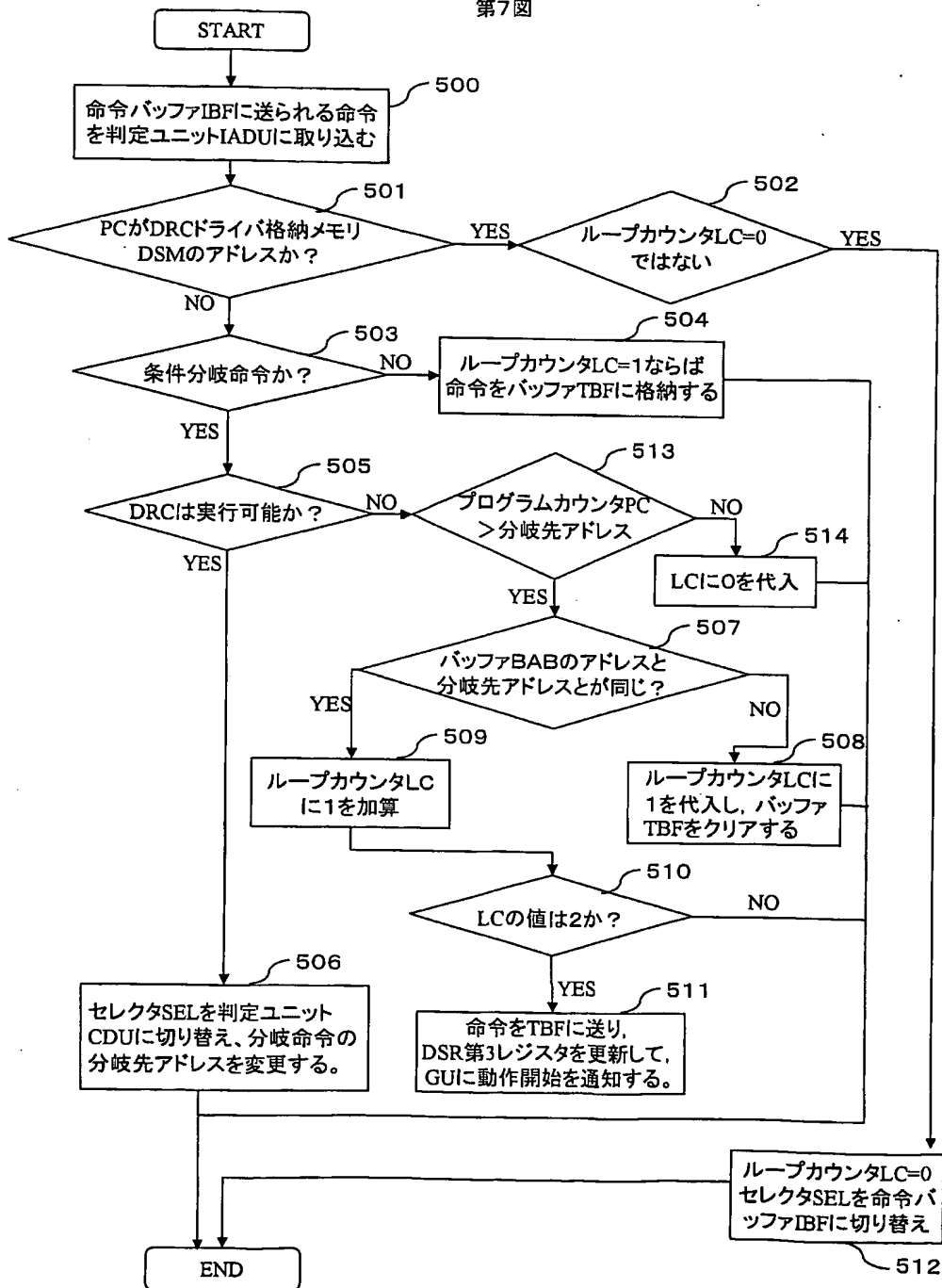
5/12

第6図



6/12

第7図



7/12

第8図

(A)

1	L001:	
2	STS	MACL,@-R15
3	MOV	#8,R0
4	L002:	
5	DT	R0
6	MOV	@R6,R3
7	MOV	@R5,R2
8	MUL	R3,R2
9	MOV	@R4,R1
10	STS	MACL,R7
11	SUB	R7,R1
12	MOV	R1,@R5
13	MOV	@R4,R3
14	ADD	R3,R7
15	MOV	R7,@R4
16	ADD	#4,R4
17	ADD	#4,R5
18	BF	L002
19	L003:	
20	NOP	
21	LDS	@R15+,MACL
22	RTS	
23	NOP	

(B)

1	L002:	
2	DT	R0
3	MOV	@R6,R3
4	MOV	@R5,R2
5	MUL	R3,R2
6	MOV	@R4,R1
7	STS	MACL,R7
8	SUB	R7,R1
9	MOV	R1,@R5
10	MOV	@R4,R3
11	ADD	R3,R7
12	MOV	R7,@R4
13	ADD	#4,R4
14	ADD	#4,R5
15	BF	L002

(C)

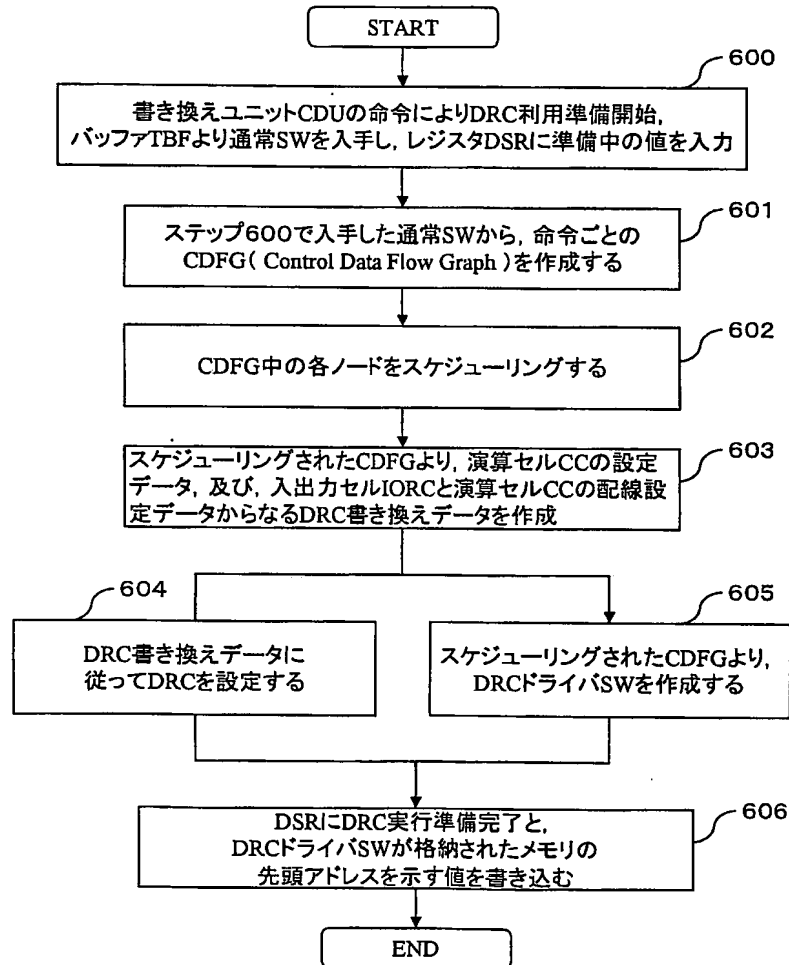
1	MOV	R6,dR6
2	MOV	R5,dR5
3	MOV	R4,dR4
4	L002a:	
5	MOV	@dR6,dR3
6	MOV	@dR5,dR2
7	MOV	@dR4,dR1
8	DT	R0
9	MOV	dR1,@dR5
10	MOV	dR7,@dR4
11	NOP	
12	BF	L002a
13	MOV	dR5,R5
14	MOV	dR4,R4
15	JMP	L003

(D)命令の意味

MOV	データ転送命令
ADD	加算命令
SUB	減算命令
MUL	積算命令
DT	デクリメント+条件セット命令
BF	条件分岐命令
JMP	無条件分岐命令
NOP	No Operation
LDS	システムレジスタロード
STS	システムレジスタストア
RTS	リターン命令
R*	汎用レジスタ*
dR*	IORC*

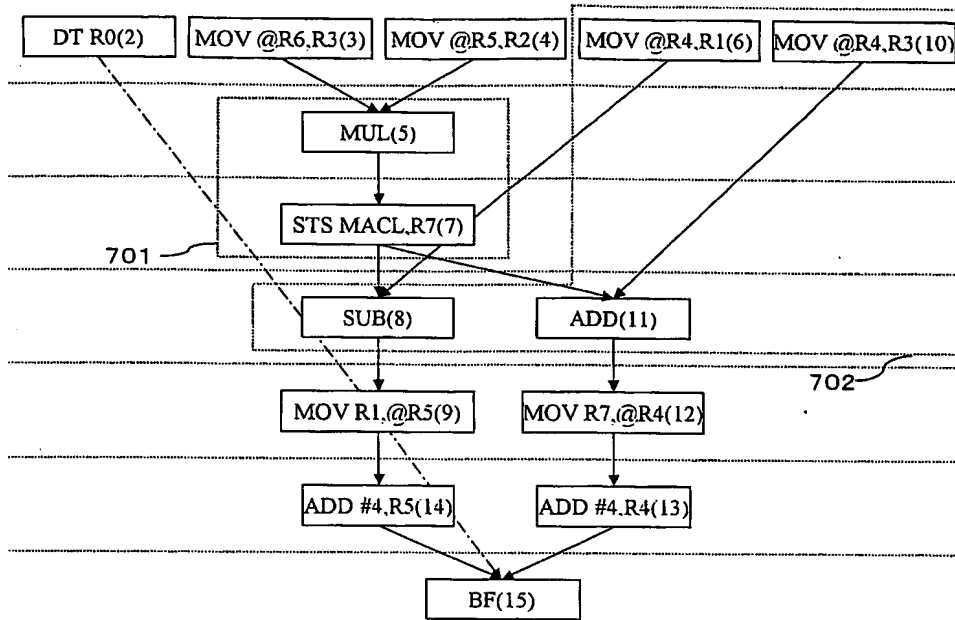
8/12

第9図

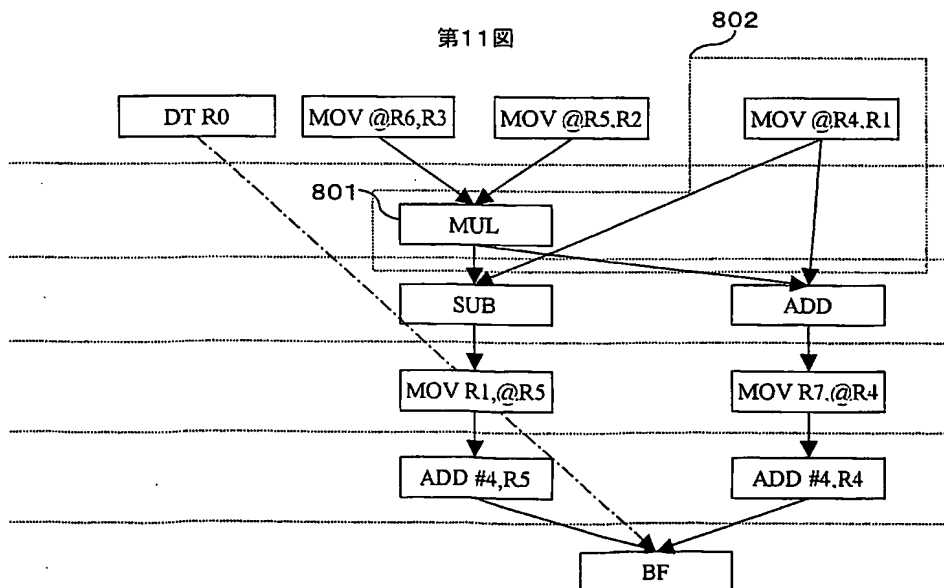


9/12

第10図



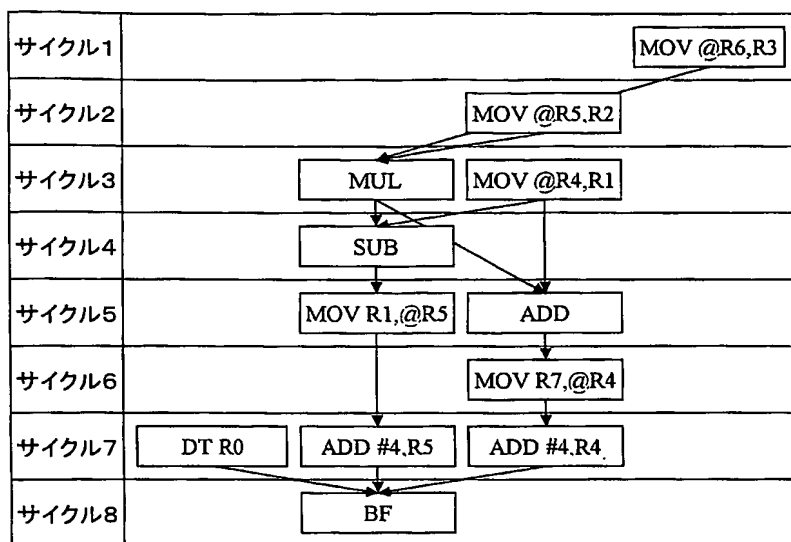
第11図



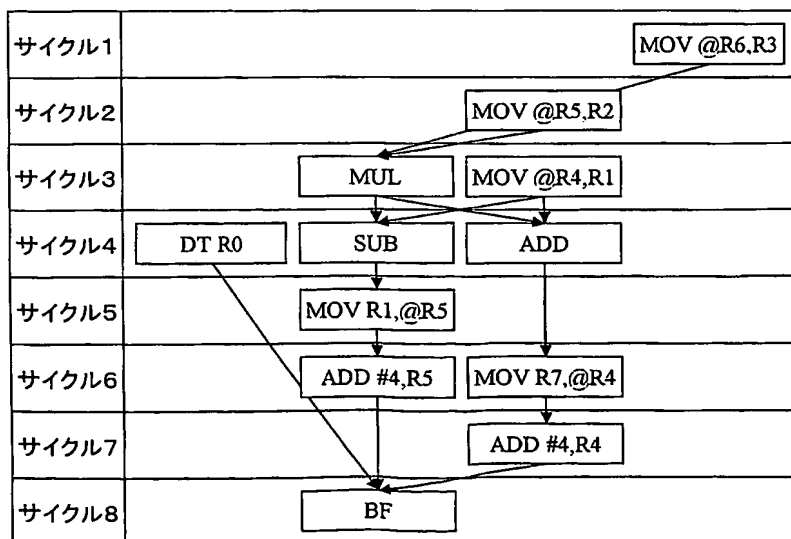
差替え用紙 (規則26)

10/12

第12図



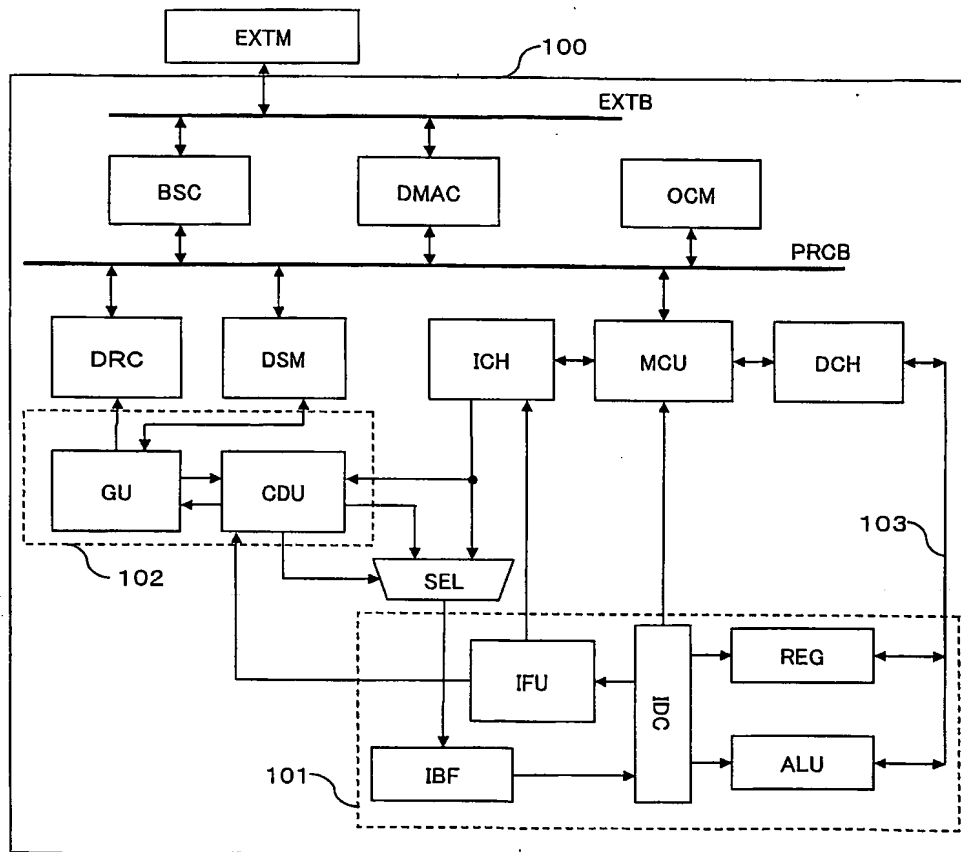
第13図



差替え用紙(規則26)

11/12

第14図



12/12

第15図

